GraphDocExplore: A Framework for the Experimental Comparison of Graph-based Document Exploration Techniques

Tobias Falke and **Iryna Gurevych**
Research Training Group AIPHES and UKP Lab
Department of Computer Science, Technische Universität Darmstadt
https://www.aiphes.tu-darmstadt.de

Abstract

Graphs have long been proposed as a tool to browse and navigate in a collection of documents in order to support exploratory search. Many techniques to automatically extract different types of graphs, showing for example entities or concepts and different relationships between them, have been suggested. While experimental evidence that they are indeed helpful exists for some of them, it is largely unknown which type of graph is most helpful for a specific exploratory task. However, carrying out experimental comparisons with human subjects is challenging and time-consuming. Towards this end, we present the *GraphDocExplore* framework. It provides an intuitive web interface for graph-based document exploration that is optimized for experimental user studies. Through a generic graph interface, different methods to extract graphs from text can be plugged into the system. Hence, they can be compared at minimal implementation effort in an environment that ensures controlled comparisons. The system is publicly available under an open-source license.[1]

1 Introduction

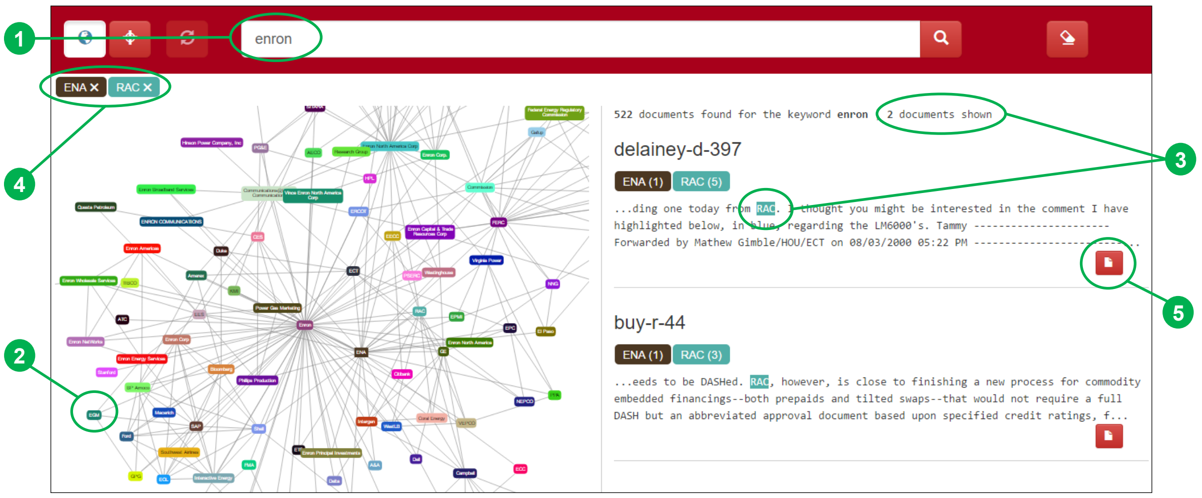
Structures that reveal relationships between different information units in a document collection, e.g. relations between mentioned organizations, have been proposed to support humans analyzing document collections. Especially in *exploratory search* scenarios, where the information need is complex and cannot be served by a simple keyword search (Marchionini, 2006), these structures are deemed beneficial. Even without supporting software, humans were found to naturally create such structures for themselves (Chin et al., 2009).

Consequently, many types of structures and approaches to extract them from text have been proposed. These include concept hierarchies (Sanderson and Croft, 1999; Yang, 2012), concept maps (Briggs et al., 2004), predicate-argument networks (van Ham et al., 2009), entailment between propositions (Adler et al., 2012) or co-occurrences of named entities (Benikova et al., 2014). All of them can be seen as labeled graphs in which nodes and edges represent different information units extracted from a document collection.

However, what remains unclear is which of these graphs are most helpful for a specific document exploration task. Only few papers evaluate their proposed graphs in a user study and usually just compare it to baselines such as keyword-based search. Direct comparisons between different types of graphs are missing.

Carrying out such a comparative user study is a difficult endeavor. Typically, one would have different groups of subjects that work on a given task under different conditions, e.g. with graph A or B. The subjects' performance on the task, measured for example in completion time or result quality, would then be compared between the groups to draw conclusions on whether graph A or B is more helpful. The first challenge is that a full end-user application has to be built around a graph-extraction method for such an experiment, which usually involves a non-trivial amount of implementation work. Second, even if full systems are already available for comparison, they might not be usable: Every difference between two systems, as small as different font sizes or colors, can influence a subject's performance. As a result, observed performance differences cannot be directly attributed to the different graphs.

[1] https://github.com/UKPLab/emnlp2017-graphdocexplore



522 documents found for the keyword **enron** 2 documents shown

delainey-d-397

ENA (1) RAC (5)

...ding one today from RAC. I thought you might be interested in the comment I have highlighted below, in blue, regarding the LM6000's. Tammy ---------------------- Forwarded by Mathew Gimble/HOU/ECT on 08/03/2000 05:22 PM ---------------------------..

buy-r-44

ENA (1) RAC (3)

...eeds to be DASHed. RAC, however, is close to finishing a new process for commodity embedded financings--both prepaids and tilted swaps--that would not require a full DASH but an abbreviated approval document based upon specified credit ratings, f...

Figure 1: User interface: After entering a search term (1), the system displays retrieved documents (right) and a graph built from them (left). When clicking on a node or edge (2), documents are filtered and highlighted (3) according to the spans associated with the selected graph element. Active filters are shown at the top (4). Opening a document (5) shows its full text with all highlighted spans.

In this work, we present *GraphDocExplore*, a framework for graph-based document exploration. Due to the following properties, it is particularly useful to carry out user studies as described above:

User Interface The framework already contains a fully implemented, modern and intuitive web application for explorative search in documents that can be used for experimental studies.

Graph-Text-Integration Rather than showing the constructed graph independently, it is tightly integrated with the documents via navigation, filtering and highlighting features to ensure that a user can effectively make use of it (see Figure 1).

Logging All actions a user performs in the web application are captured in a detailed log for further analysis and reconstruction of user behavior.

Graph-Independence Different methods to extract graphs can be plugged into a generic interface, such that fair experimental comparisons that effectively control for all other confounding factors can be set up easily and quickly.

Dynamic Graphs Integrated methods for graph extraction are notified about all user actions and can dynamically modify their graph during a session, allowing to study the personalization of a graph based on the user actions.

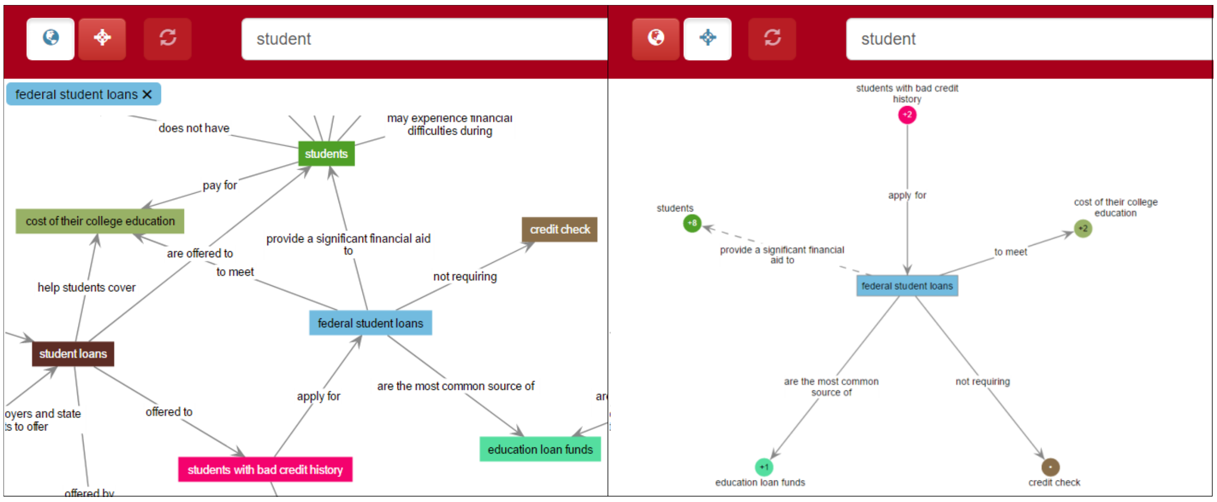
The remainder of this paper is organized as follows: First, we review different types of graphs and corresponding systems proposed in the past (§2). Then, we present our framework from a functional (§3) and technical perspective (§4). Finally, we report results of a first user study (§5).

2 Related Work

Many methods to structure document collections can be seen as different kinds of labeled graphs generated from the documents. Early work studied *concept hierarchies* which are graphs with concepts as nodes and edges representing hyponomy relations (Sanderson and Croft, 1999; Lawrie et al., 2001; Kummamuru et al., 2004). More recently, personalized versions were proposed (Yang, 2012, 2015). *Concept maps* are a more expressive variant of these graphs in which the edges have different labels defining their meaning rather than all being taxonomic (Novak and Gowin, 1984) and can be used for the same purpose (Briggs et al., 2004). Another popular type of graph shows *keywords* (Tixier et al., 2016) or *entities* (Benikova et al., 2014) as nodes with unlabeled edges between them depicting co-occurrences. Other graphs were suggested to depict entailment (Adler et al., 2012) or relations expressed by a specific predicate (van Ham et al., 2009). All of these structures can be captured with the abstract graph model in our framework.

In addition to specific graphs, more complex applications, such as *new/s/leak* (Yimam et al., 2016), *Jigsaw* (Görg et al., 2013) or *Overview*[2],

[2]https://www.overviewdocs.com



student
federal student loans ✕
does not have
may experience financial difficulties during
students
pay for
cost of their college education
provide a significant financial aid to
credit check
are offered to
to meet
not requiring
help students cover
federal student loans
student loans
are the most common source of
apply for
oyers and state s to offer
offered to
education loan funds
students with bad credit history
offered by
student
students with bad credit history
apply for
students
cost of their college education
provide a significant financial aid to
to meet
federal student loans
are the most common source of
not requiring
education loan funds
credit check

Figure 2: Layouts: *Full (left):* The graph is displayed completely and can be zoomed, panned and moved. *Focused (right):* A single focus node and its neighbors are visible. Selecting a neighbor moves the focus.

have been developed to support document exploration by integrating many different techniques, including graphs, in a single application. In contrast to our work, their focus is on productive use, while we are mainly interested in the experimental evaluation, requiring us to study the use of specific graphs in isolation. Moreover, note that graphs in which nodes represent full documents, as obtained by traditional document clustering or document chains (Shahaf and Guestrin, 2010), are less useful for our application because no fine-grained highlights can be provided for nodes and edges.

With regard to experimental evaluations, only few of the suggested graphs were extrinsically evaluated and proven to be helpful in an exploratory search scenario. Both Kummamuru et al. (2004) and Yang (2012) compare their approaches with previous work in user studies, but restrict the comparison to other methods producing concept hierarchies. Kang et al. (2011) compare the powerful Jigsaw system against simpler alternatives, including keyword-based search and pen and paper. To the best of our knowledge, no studies have been carried out for other types of graphs, e.g. for concept maps, and there are no studies comparing different types of graphs against each other. Thus, the framework presented in this work is highly needed to make such comparisons.

3 Functionality

The system for graph-based document exploration was designed in the style of well-known search engine interfaces. As shown in Figure 1, the list of search results is complemented by a visualization of the graph that has been extracted from the retrieved documents. This type of integration follows the popular paradigm of faceted search, in which different taxonomies, either predefined or extracted from the results (Hearst and Stoica, 2009), are offered along with the results to filter them. Instead of the typically small, single-level taxonomies, our application takes this idea further by offering a comprehensive graph.

User Interaction After executing a query, a user can both scroll through the list of retrieved documents or navigate through the generated graph. Every node and (optionally) edge in the graph is associated with at least one span in one of the retrieved documents (see Figure 5). Note that these spans must not match the label of the graph element, but can be other phrases referring to it. If a user selects an element in the graph, the corresponding spans are highlighted in the document snippets and the results are filtered to the subset of documents that contain at least one associated span. Filters can be combined, which reduces the documents to those containing spans for all, can be temporally deactivated and can also be removed completely. In the result list, the number of corresponding spans for each filter is displayed. Colors of nodes in the graph, filter tags and highlighted spans match. In the result list, a user can switch to the full text view of a document, which also contains highlights according to the current filters.

Graph Layouts The application currently provides two different graph visualizations that we found to yield useful renderings for graphs of different sizes. At any point, a user can switch between them using the buttons in the top left corner. The left part of Figure 2 gives an example of the *full* layout, which is a force-directed layout showing the complete graph. It allows the user to zoom in and out and pan to fully inspect the graph. While it has the advantage that it can provide an overview of the complete graph, the visualization can become complex for large graphs.

As an alternative, the application offers a *focused* layout, which is shown on the right side of Figure 2. It shows only one focus node and its direct neighbors at a time, while every neighbor node has a number indicating how many more edges are connected to it. Selecting one of the neighbors moves that node to the center and displays its neighbors. This allows to go through the graph step by step and it usually yields much cleaner visualizations, as the number of visible nodes is limited. Both visualizations support directed and undirected graphs. The modular design of the application allows to add alternative visualizations in the future.

Logging In order to be able to thoroughly study the behavior of users that are working with the application, it creates a comprehensive log of all actions that a user performed. Figure 3 illustrates how such a log looks like for a user session. The user, working on documents about student loans, starts by issuing the query *credit check*. The corresponding log entry lists the keyword and a list of the retrieved documents (1). Next, she scrolled through the result list and stopped with documents 3, 26 and 17 in the visible section of the list (2). Zooming the graph made a certain set of nodes visible (3). She then selected one of the nodes (4), which automatically created a filter (5) and reduced the result list correspondingly. From the filtered result list, she opened document 17 (6), scrolled to a certain position (7) and closed it afterwards (8). She then switches to the alternative graph layout (9) and continues her search. Note that the actual log also contains timestamps.

```
1 SEARCH “credit check” doc-0,doc-8,doc-3,...
2 RES_SCROLLED doc-3,doc-26,doc-17
3 ZOOMED 42,65,89,57,35,24
4 NODE_CLICKED 24
5 FILTER_ADDED 24
6 DOC_OPENED doc-17
7 DOC_SCROLLED [516-1468]
8 BACK_TO_RES
9 GRAPH_SWITCHED focused
...
```

Figure 3: Example for a user interaction log.

4 Architecture and Implementation

The framework has been implemented following a server-client-architecture and designed to be easily extensible in different regards. In the following, we provide an overview of the architecture and describe two aspects, the integration of graph generators and dynamic graphs, in more detail.

Overview Figure 4 depicts the architecture of the framework. The server-side portion is realized in *Java*. To enable the keyword search, we integrated *Apache Solr*[3] to index the documents. Different graph generation modules can be plugged into the system and have access to the documents. Several document collections can be loaded into the framework and used with different graphs. User actions are logged in a text-based format. In addition to the graphs, the modular design also allows to easily change the search engine, e.g. to Lucene, or the logging mechanism, e.g. to a database. On the client-side, the user interface described in the previous section is realized with *Angular JS*.[4] The server exposes a *REST API* to handle all communication with the frontend.

Graph Interface As mentioned above, different graph generation modules can be used with the framework. A configuration file defines the active type of graph per document collection. When starting a new search, the framework instantiates the corresponding graph generator and provides the retrieved documents. The generator can then apply its custom processing logic to the documents and return the resulting graph. To offload expensive preprocessing work, a generator can also access precomputed data for each document. Figure 5 shows the data structure in which a graph is represented in the framework. Both a node and an edge have a label, containing a description string used in the visualization and a list of spans in the documents. Note that the latter is crucial for the interaction between the graph and the documents through filtering and highlighting. The sys-

[3]http://lucene.apache.org/solr/
[4]https://angularjs.org/

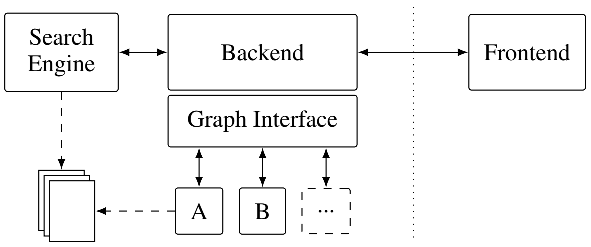


Figure 4: System architecture, enabling the integration of different graph generation models.

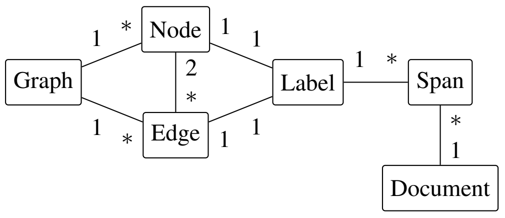


Figure 5: Data structure to capture different types of text graphs in UML-style class notation.

tem supports both labeled and unlabeled as well as directed and undirected graphs.

As examples, we created two graph generation modules for our framework. The first is a file-based generator that simply reads a static graph for a document collection from a file. The second integrates DKPro Core (Eckart de Castilho and Gurevych, 2014) to create co-occurrence graphs of automatically recognized entities. It demonstrates how a broad range of linguistic preprocessing tools can be easily made available and utilized in our framework. Similarly, many other graph generation methods can be used with the framework. When comparing them against each other in a user study, the common web application ensures a controlled experimental setting.

Dynamic Graphs As part of the graph generation interface, implementations are also notified about the actions that a user performs in the application. All events that are documented in the interaction log (see Figure 3) are provided through the interface. Further, a graph generator can modify its generated graph after the initial creation. In case of a change to the graph, the user is notified and can trigger an update of the visualization in the frontend. This setup makes it possible to create dynamic instead of static graphs and to use the framework to study their usefulness. While some work has been done in this direction (Yang, 2012; Shahaf and Guestrin, 2010), the development of methods that interactively adapt and personalize a graph for document exploration to a specific user has received only little attention. With our application, we provide an important evaluation framework that is needed to move further into this interesting direction of research.

5 User Study

To verify whether the user interface and interaction of the presented application is in line with user expectations, we conducted a first preliminary user study. Since the focus was on usability, all subjects worked with the same graph rather than comparing different graph types. 20 researchers from our lab and students from the university participated. They used the application to explore a collection of web pages on student loans (as in Figure 2) and answered a questionnaire asking for feedback on different parts of the application.

The results showed that the application was perceived as being very intuitive. Subjects could easily interpret the meaning of the graph and how it can be used to filter and highlight the documents. With regard to the different layouts, 60% preferred the focused layout because it was “clearer” and “less cluttered”, while only 15% preferred the full layout, the rest being undecided. However, several subjects noted that the full layout is still useful to get the big picture, advocating to offer both options in the tool. In addition, the participants provided many useful suggestions to improve the application, e.g. adding tooltips, which have been incorporated into the current version.

6 Conclusion

In this paper, we presented *GraphDocExplore*, a framework for graph-based document exploration. Its web application augments a traditional keyword-search interface with a graph extracted from the search results. The graph can be used to navigate, filter and explore a collection of documents in an intuitive way. With its generic graph generator interface, different approaches to extract graphs from text can be plugged into the framework, providing an ideal environment to compare these approaches in controlled experimental evaluations with users. Further, the framework supports graphs that are dynamically altered based on user interactions, allowing to study methods for the interactive personalization of navigation graphs. The framework and its source code are publicly available at `https://github.com/UKPLab/emnlp2017-graphdocexplore`.

Acknowledgments

We would like to thank Alexander Gerhard Gössl, Arwed Gölz, Ramy Hcini and Christoph Sebastian Vollbrecht for their help with the implementation and all participants of the user study for their feedback. This work has been supported by the DFG as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1.

References

Meni Adler, Jonathan Berant, and Ido Dagan. 2012. Entailment-based Text Exploration with Application to the Health-care Domain. In *Proceedings of the 50th Annual Meeting of the ACL*, pages 79–84, Jeju, Republic of Korea.

Darina Benikova, Uli Fahrer, Alexander Gabriel, Manuel Kaufmann, Seid Muhie Yimam, Tatiana von Landesberger, and Chris Biemann. 2014. Network of the Day: Aggregating and Visualizing Entity Networks from Online Sources. In *Workshop Proceedings KONVENS 2014*, Hildesheim, Germany.

Geoffrey Briggs, David A. Shamma, Alberto J. Cañas, Roger Carff, Jeffrey Scargle, and Joseph D. Novak. 2004. Concept Maps Applied to Mars Exploration Public Outreach. In *Concept Maps: Theory, Methodology, Technology. Proceedings of the First International Conference on Concept Mapping*, pages 109–116, Pamplona, Spain.

George Chin, Olga A. Kuchar, and Katherine E. Wolf. 2009. Exploring the analytical processes of intelligence analysts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 11–20, Boston, MA, USA.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland.

Carsten Görg, Zhicheng Liu, Jaeyeon Kihm, Jaegul Choo, Haesun Park, and John T. Stasko. 2013. Combining Computational Analyses and Interactive Visualization for Document Exploration and Sensemaking in Jigsaw. *IEEE Transactions on Visualization and Computer Graphics*, 19(10):1646–1663.

Marti A. Hearst and Emilia Stoica. 2009. NLP support for faceted navigation in scholarly collections. In *Proceedings ofthe 2009 Workshop on Text and Citation Analysis for Scholarly Digital Librarie*, pages 62–70, Singapore.

Youn-Ah Kang, Carsten Görg, and John T. Stasko. 2011. How Can Visual Analytics Assist Investigative Analysis? Design Implications from an Evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):570–583.

Krishna Kummamuru, Rohit Lotlikar, Shourya Roy, Karan Singal, and Raghu Krishnapuram. 2004. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th International Conference on WWW*, pages 658–665, New York, USA.

Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. 2001. Finding topic words for hierarchical summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference*, pages 349–357, New Orleans, LA, USA.

Gary Marchionini. 2006. Exploratory Search. *Communications of the ACM*, 49(4):41–46.

Joseph D. Novak and D. Bob Gowin. 1984. *Learning How to Learn*. Cambridge University Press, Cambridge.

Mark Sanderson and Bruce Croft. 1999. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference*, pages 206–213, Berkeley, CA, USA.

Dafna Shahaf and Carlos Guestrin. 2010. Connecting the Dots Between News Articles. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 623–632, Washington, DC, USA.

Antoine Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. 2016. GoWvis: A Web Application for Graph-of-Words-based Text Visualization and Summarization. In *Proceedings of ACL-2016 System Demonstrations*, pages 151–156, Berlin, Germany.

Frank van Ham, Martin Wattenberg, and Fernanda B. Viegas. 2009. Mapping Text with Phrase Nets. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1169–1176.

Hui Yang. 2012. Constructing Task-Specific Taxonomies for Document Collection Browsing. In *Proceedings of the 2012 Joint Conference on EMNLP and CoNLL*, pages 1278–1289, Jeju Island, Korea.

Hui Yang. 2015. Browsing Hierarchy Construction by Minimum Evolution. *ACM Transactions on Information Systems*, 33(3):1–33.

Seid Muhie Yimam, Heiner Ulrich, Tatiana von Landesberger, Marcel Rosenbach, Michaela Regneri, Alexander Panchenko, Franziska Lehmann, Uli Fahrer, Chris Biemann, and Kathrin Ballweg. 2016. new/s/leak – Information Extraction and Visualization for Investigative Data Journalists. In *Proceedings of the 54th Annual Meeting of the ACL*, pages 163–168, Berlin, Germany.